**Module 4 - A103**

**Introduction to ITS Standards Requirements Development**

**Webinar Transcript**

*Shelley Row:*

ITS Standards can make your life easier. Your procurements will go more smoothly and you'll encourage competition, but only if you know how to write them into your specifications and test them. This module is one in a series that covers practical applications for acquiring and testing standards-based ITS systems.

I am Shelley Row, the director of the ITS Joint Program Office for USDOT, and I want to welcome you to our newly redesigned ITS Standards Training Program of which this module is a part. We are pleased to be working with our partner, the Institute of Transportation Engineers (ITE), to deliver this new approach to training that combines web-based modules with instructor interaction to bring the latest in ITS learning to busy professionals like you.

This combined approach allows interested professionals to schedule training at your convenience, without the need to travel. After you complete this training, we hope that you will tell colleagues and customers about the latest ITS Standards and encourage them to take advantage of the archived version of the webinars.

ITS Standards training is one of the first offerings of our updated Professional Capacity Training (PCB) Program. Through the PCB program, we prepare professionals to adopt proven and emerging ITS technologies that will make surface transportation safer, smarter, and greener, which improves livability for us all. You can find information on additional modules and training programs on our web site: www.pcb.its.dot.gov.

Please help us make even more improvements to our training modules through the evaluation process. We look forward to hearing your comments. Thank you for participating and we hope you find this module helpful.

*Announcer:*

Welcome to Module A103, Introduction to ITS Standards Requirements Development. The target audiences are decision makers, project managers, operational stakeholders. Your instructor is Ralph Boaz, President of Pillar Consulting, and is an engineering, management, and transportation consultant with 29 years of experience leading and developing successful commercial, government, and research projects and products. The next voice you will hear will be of your instructor.

*Ralph Boaz*

It's a pleasure to be with you. We're going to have an exciting class. A lot of material will be presented, and I just would like to ask that you all be on your toes because I like to have an interactive class, and so you'll be using your Chat Box quite a bit, I hope, and you can enter questions as I speak, or you can answer my questions as they come along through the presentation. And hopefully I will get to any questions you ask at the end different sections in the module. But if not, then we'll re-ask questions at the end of the course. So with that we'll begin.

I like to ask this question, because it's very enlightening. So who are stakeholders? How many would say that a TMC operator, a field maintenance person, or operational support—would you say—are stakeholders in a system? I think that most everybody would say yes, that this in fact the case, that these are the people we normally think about when we are building a system. These are typically people that are closely associated and working together.

How about interfacing system owner? You'd be surprised about how many people don't include an interfacing system owner or a purchaser of the equipment. But they, too, are stakeholders in the system. And what about the sponsor? Now, briefly what a sponsor is—this is the champion for your project, and every successful project has a sponsor, somebody to protect it. It may be at various levels of an organization, but it's somebody who has influence to make sure that the project goes forward. They are a stakeholder.

Now, anybody want to take a guess about this one, a regulatory agency? That might be something like somebody who delivers radio frequency bands, or it might be also somebody who allows railway access. Does anybody want to take a guess on this one? Are they a stakeholder? Yes, we got a yes, and absolutely, they are stakeholders. Not all projects have this kind of thing but if they are there, they are stakeholders.

Okay, now here's the biggie. Here's the last one. Is the public or is a politician a stakeholder? Well, I got an "absolutely" on that one. Yes. Often on projects, we talk about the public maybe as a beneficiary, but depending on who was the instigator of a project—like, for instance, a community that demanded a traffic signal or that traffic safety equipment be installed—they become stakeholders in that case, not just the beneficiaries. And in the case of a politician, a politician is a stakeholder if he made promises—for instance, in his campaign, about improving traffic or things like that—that makes him not just a beneficiary of it but he's also a stakeholder. And sometimes the politician is the sponsor of the project, so it depends on the project whether the public or politician are stakeholders or whether they're beneficiaries, but we also want to include them in our list.

Now, something that's helpful to look at is what we call an onion model, and you see some of the stakeholders more central to this diagram and others on the outside of the layers of this "onion." When we speak about operational stakeholders, we're talking about these people who are very close to the physical system, which is what normally people would think – operational support, field maintenance, traffic management center operators, etc. The interesting thing maybe is the next level out, the interfacing system owner, and I've provided courses and taught all over the country in the last decade, courses on traffic and system engineering, and I've found that many

times the interfacing systems between adjacent cities, they have no idea, they've never spoken to the people except once they got into the class they started talking to each other. So it's really important that we think outside our initial purview and outside of our groups, or outside of the funded group, for instance, on a project, because there are a lot more people at stake.

And one of the things that's really important, I think, is the concept of an integrated project team. What I mean by that is it includes people with all the skills and knowledge for the project to succeed. Now, you can say, "Well, that's a huge group." However, you don't need everybody to be participating all the time. You just make sure that there's good communications, and when you need them they're accessible.

Do I have any questions about stakeholders? Okay, we'll move along. Here you see our curriculum path. We call this non-SEP, SEP standing for Systems Engineering Process, and we'll get more into that. You probably got quite a bit of that in the other courses, but this path is intended to support all of you and provide you with information to go through the use of standards that don't contain Systems Engineering Process information, which is helpful when you're creating a system. But here are our courses. It's using ITS Standards, an Overview; Introduction to Acquiring Standards-based ITS Systems; Introduction to User Needs Identification; Details on Acquiring Standards-based ITS Systems; Identifying and Writing User Needs When ITS Standards Do Not Have SEP Content; and then this course which is Introduction to ITS Standards Requirements Development.

Then we'll have a course that will follow this and it's very closely related. A203, which you can take later, is Writing Requirements When ITS Standards Do Not Have SEP Content, and then we'll go to have some courses on Identifying and Writing Specific User Needs for various courses, and these are expected in the next couple of years. I'm actually not sure of the time frame on that, but we know that you'd like some help on specific standards. I would say that I would recommend all of these courses, but that hopefully by now you have taken A102, Introduction to User Needs Identification.

This is just the same thing as the previous slide so I'll skip that. I know that with the people on this phone call that you have varying degrees of experience, and some of you may be experts in some of these areas. If you have knowledge in any of these areas, it is helpful, and some of the material we'll receive today will be maybe review or old news, but I'm sure that there will be parts of it that will fill in some gaps and holes and connect it to the other portions of the course. So we have Intelligent Transportation Systems, managing ITS deployment projects, government procurement processes, the benefits of standards, the Systems Engineering Process in any of these areas is very helpful.

We will be having six learning objectives and these are the first three. So at the conclusion of the module you will be able to define requirements for overall operation to satisfy user needs, understand the concept of a well-formed requirement, and to define the system and interfaces as a functional architecture. You'll also be able to use decomposition of the architecture and requirements as necessary to properly define the system. You'll be able to verify that requirements are complete and correct, and understand how requirements developments apply to ITS communication standards.

The big thing here is—I'm going to back up a slide—on this first point, you need to understand that the problem is bigger than you think. When we talk about well-formed requirements, we're going to talk about the structure for expressing requirements, and it's a concept that was developed by NASA. And then we'll talk about system boundaries. When we're talking about the functional architecture we'll be talking about system boundaries and interfaces, where you'll see where a picture is worth 1,000 words, as they say. This is important to give context to the requirements.

In decomposition, we'll see that, hey, when you think about it at a high level, that's maybe not to the level that is needed to properly define the system, so that's what decomposition does. And this next item, verifying requirements, we basically don't want to get to the end of the project and find out that we didn't get what we wanted or we didn't express what we wanted, or we expressed it incorrectly and got what we wanted, and you see my point there. And again, that last one is, we'll just see how all of what you learn today applies to ITS communication Standards.

Now, the emphasis of the course will be on requirements in general, and then in this last part of the course we'll bring it into applying it to the standards themselves. And these points, we're going to use these learning objectives as sections of the course and tie it all back together in the end.

So this is our learning objective number one. In looking at overall operation to satisfy user needs, we're going to review the system's life cycle—and I promise to be brief on that since you've seen that—review concepts of operations and definition, and the same goes there. We'll discuss the relationship of user needs to requirements. Needs and requirements are closely related but they're different; we'll take a look at that. We'll take a look at the role of requirements in terms of the overall system life cycle. So that leads us into the review of the systems life cycle, or we call this the V or the Flying V because of the horizontal parts at the top of the V.

And I want to be clear about something—this has been confused in many presentations—that this does not represent the Systems Engineering Process, but this represents the systems life cycle. The Systems Engineering Process may take on various forms, like there may be a waterfall model of development; there may be a fast prototyping approach. But I will say this: I'll venture to say that all successful systems will go through these stages of development one way or another. Before they're done and successful, you will find that they've gone through these steps.

This formalization, I'm going to give you a definition for systems engineering. This comes from the International Council on Systems Engineering. It's an interdisciplinary approach and means to enable the realization of successful systems. The use of systems engineering is advocated throughout the standards programs. The principles are the underpinnings of the system life cycle diagram that you see everywhere, and we're going to focus on the development portion of the system life cycle. This is the part that goes from maybe the concept of operations down and up the V to system validation.

So we have, then, the concept of operations. This is where we define user needs. From the user needs we develop system requirements. From the system requirements we get high-level design. And then we go down to detailed design, and depending on what kind of project we have, we'll

develop software or hardware or, in the case of an integration project, we'll be maybe making field installations. We call this an implementation part. Then we do some unit or device testing, lower level testing. Then we have subsystems of our large system being verified, and then we have the overall system verified and complete our deployment. And then we can see that the system is validated. We'll go through a validation point, just to go back, and we validate that we have met those user needs that were identified in the concept of operations.

Let's talk a little bit about the components of a concept of operations. I took this example from the FHWA Systems Engineering Guidebook. It is one of many outlines that can be used for a concept of operations. There are other well-known ones, like, for instance, IEEE Standard 1362, and there are usually some rules or some suggestions on customization. ConOps might be a standalone document or the concept of operations might be part of another document, such as a requirements document or, in our case, the standard, the concept of operation ends up going in as part of the standard.

One of the other points I wanted to make—here are some other sections that you can see—in a Systems Engineering Guidebook it has operational needs, system overview, operational environment, support environment, operational scenarios, summary of impacts, and these things were probably covered in the previous course. However, I want to make a point that some of the outlines that are used won't actually have a section that says "these are the user needs," so you will usually find the appropriate place in the outline that's provided and then it's recommended that you do have a section there that you identify those user needs.

You will have covered this in the previous course, and we're going to talk about characteristics of well-written user needs. They are uniquely identifiable, they take on a major desired capability, they are solution-free. Many times when we have workshops and seminars, users present their solution for something that they really needed done, and it takes a little bit of asking questions to find out what the real need was, because when you're starting a project and doing a system a project, you don't want to define what the end product is going to be or the design is going to be. You want to leave that open as you go through the Systems Engineering Process. So you want to express your user need in a solution-free fashion.

And the last thing here, captures rationale, is probably the most important thing out of this whole list, because sometimes we don't express our user needs in the best way, but if we put a rationale there, we can always go back and look and see what we meant. Often, in the next stages of development of a system—for instance, in the requirements development—you find out that, oh, that user need was really too specific; we need to revise that. Or people assume what that was by just the title and then you go back and you take a look at the rationale and you see that some of the requirements that were created weren't really capturing what the rationale was from the previous stage of development.

Here we have an example user need, and here's your opportunity to show off. Can you tell me what makes this user need uniquely identifiable? I'll give you just a few more seconds. Okay, well, this should have been an easy one. It's uniquely identifiable because we've given it an identifier, in this case 4.3.1.11. So this would be a way of expressing a user need and uniquely identifying it. What's the major desired capability? You can just use your chat box down there to

answer. Well, I'll help you with this one also. It's “limit audible noise.” So we have it uniquely identified and then we have a major capability that users want to be able to limit audible noise.

Now, here is the user need. It says, “the user needs the TFCS.”—now what the TFCS stands for is Transportation Field Cabinet System—“to have limited audible noise.” So they don't want a traffic control cabinet to be making a lot of noise. Now, in this “the user needs the TFCS to have limited audible noise”—is it solution-free? Has any solution been expressed in that? Oh, this is a good question. Somebody asks, “What about ped buttons?”. Well, yeah, because sometimes we have ADA and other kinds of things where we want countdown peds, etc., but we're not going to get into the details of whether that should be in our system or not, because that's outside the scope and I'm just going to limit this to this particular example. But it was a good point. And there's no solution. Thank you. It's stated just what they wanted.

And then we take a look at the rationale. “The TFCSs will be deployed in areas where residents are sensitive to ambient sound.” That's why we had this user need. I'm going to give you a hint on this one. “It is said that user needs identify the high-level blank of the system?” and the hint is, is this a who or is this a what or is this a how? A high-level who, what, or how of the system? That's correct. It's the what. We're trying to understand what we are trying to build.

Now we're going to take a look at the relationship of user needs to the requirements, and here's a definition of a requirement, and I'll just read the underlined portions. “Translation of needs into a quantified or descriptive specifications to enable its realization,” and that comes from the ISO/IEC Guide 25. And you can see all these references are probably in detail referenced in your Student Supplement, which has some other materials in it that I encourage you to refer to offline. It may have some answers to some of the questions I have in it, so I hope we can just focus on the presentation.

What we're really saying is the requirement is more concrete than the need, right? That's really what we're saying. Let's look at that user need that we had before. Now, sometimes user needs, when we talk about this, we talk about user needs as goals, and the goals are not something that would be nice if we get there, but they are something we have to achieve to win. So the idea of a user need as a goal is we have to achieve this to win. So here is our user need: “The user needs the TFCS to have limited audible noise.” Now look at the requirement. How much more concrete is this—that we have requirement 5.1.20, Audible Noise Level: “The TFCS shall have no component that emits an audible noise level exceeding a peak level of 55 dBA when measured at a distance of one meter away from its surface.” You see that the requirement is concrete, it's clear.

Looking more at our relationships of user needs to requirements, we see in our documents or in our systems development that a need will sometimes apply to one requirement. You'll only have one requirement that really meets the need that was expressed, and we call that a one-to-one relationship. You see this quite a bit. This is a one-to-many relationship where we have a user need expressed and it takes multiple requirements to properly express that user need in a fashion that a requirement is concrete, attestable, and some of the other subjects we're going to talk about in a minute.

And often a user need has different aspects to it that really don't get ironed out until they're expressed as requirements. Now, if we have a one-to-one and we have a one-to-many, what would be the next one? What's the next relationship we have? Anybody want to take a shot? Many-to-one. Very good. Thank you. This is the advanced class, I can tell. This is a case where you have multiple user needs but one requirement really satisfies them, and that often occurs when you have a user need that maybe refers to the whole system, like maybe it's a user need that has to do with quality or the like. Some general type of user need—it applies to many of the requirements that you have—so that's often the case where you get a many-to-one relationship, but not solely. Sometimes there will be a specific need expressed that the same requirement answers those two needs. In practice, all of these relationships will be occurring in the same document.

And this next slide is just to emphasize the importance of requirements. At a system level, requirements are the key to acceptance. They help you determine success. There's a whole other subject that we won't go into right now that's called a requirements-driven testing, and it all has to do with expressing those requirements and then being able to automate some of your testing based on that, and it's a subject in itself.

Now, there are different types of requirements. We have functional requirements, we have performance requirements, we have non-functional requirements, and we have what we call architectural constraints, or simply constraints. Functional requirements, when we talk about them, we're talking about what the system shall do. When we're talking about performance requirements, we're talking about how well the system should perform. And under non-functional requirements, we're talking about under what condition the system shall perform. Example: reliability, safety, environment, security, training. There's a whole list in your Student Supplement. It's just about anything that's not covered as the primary capability of the system. And under architectural constraints or constraints, that's anything that constrains the development. Sometimes that will be technology, it will be equipment, design. There will be some tools or standards being used. Architectural constraints apply at the highest level. There may be other constraints discovered as the systems decompose. A good example is if an agency already has a wireless communications infrastructure, for use by the agency, I should say. So there may be an architectural constraint that a system going in needs to use that wireless infrastructure.

Let me ask this. Are there any questions about what we've learned so far? Okay. Now we'll go into talk about our next learning object, which is the concept of a well-formed requirement, and in doing that we're going to talk about the structure of a well-formed requirement, and secondly, we'll talk about the characteristics of a well-formed requirement. Again, as I said previously, this idea first came from NASA and it has the following parts: actor, action, target, constraint, and localization.

The actor, this is typically the system or subsystem or name thereof that identifies what does the action. The action identifies what is to happen, and that's usually expressed as a "shall" statement. "The system shall *blank*." The target identifies what receives the action, and the constraint and localization; they are the things that really make it mean something. The constraint is how to measure success or failure. Localization identifies the circumstances under

which the requirement applies. Now, it should be noted that both the constraint and localization really make it mean something. They're important but not all requirements will have both.

We'll just step through this. Not all requirements have to be expressed this way. I've seen good documents that have them expressed differently. But if you cannot express it this way, it's likely that it needs to be simplified. So we have the system which, again, is our actor, shall generate, which is our action, the event reports, which is our target, containing the following information—which isn't expressed here. It might be under a sub-paragraph or something—but containing the following information, which is our constraint, on a scheduled interval, which is a localization. And as I said, if the requirement can't be stated in this simple format, you probably need to define the functionality using multiple requirements.

And again we have a frozen screen here. Just a moment. I'll say this: I have written documents where I did not express requirements in this fashion, but I really appreciated, in working with the USDOT on their project, that they like it this way, but I have become a big fan of this. And also is that it's easy for users to get used to the style, and the key items are easy to pick up if you're consistent with this.

Now we'll look at characteristics of a well-formed requirement. The first one is it must be necessary. How do you know that? Well, it's useful and traceable to needs. It's concise. We want it to be minimal and understandable and expressed in a declarative language, like "shall statements." We want it to be attainable. Sometimes someone will express what they think is a requirement but it will be something that you can't really achieve. An example might be "the system shall have a life expectancy of 20 years." Now, that might be okay as a need, without the shall statement, but you need to have something in a requirement that you can actually test, and you won't wait 20 years to test that particular requirement.

It must be standalone, so the requirement needs to be stated completely in one place. It needs to be consistent, and by that we mean it shouldn't contradict itself, nor any other requirements. And unambiguous, we only want one interpretation. And then we say it's verifiable, and that means it can be met through inspection, analysis, demonstration, or test. So those are the ways that we verify that the requirement can be met. If we can't inspect it, do analysis, demonstration, or test it, then we probably have something that goes back to being unattainable.

Here we have an example requirement, and I'm going to give you the first one. We want to find out what is the actor in here, and I'm giving you this one. It says "The TFCS," and it says "no component," could have been expressed "no component of the TFCS" but I wanted to keep with our style. So we'll say that this is our actor. Now what I'd like to do is ask you all to tell me what the action is. Emits. Very good. Emits is our action. Then we want to look at, okay, what's our target? One of you had kind of covered it in your previous answer. Yes, audible noise. So audible noise level is our target. Then we want to look at the constraint, so what's the constraint in this case? Right. Exceeding a peak level of 55 dBA. Very good. And then what's our localization? Right. When measured at a distance of one meter away from its surface.

So this looks pretty. This looks like a pretty well-formed requirement. Now let's go ahead and take a look at the some of the other features here. Is it necessary? Well, we know it's necessary—

well, the comment of maybe. Right. Maybe is because if we take it out of context we don't know what the user need was, but assuming that we have the user need that we expressed earlier in this presentation, we could say that, yeah, it's necessary. It's traceable to a need. Is it concise? It's expressed in the declarative language and it's understandable, so I'd say yes, it's concise. Is it attainable? Are we able to achieve this in the available resources and time? Yes. Stand alone? It's here. It's all in one place.

Let me ask you this one. Is it consistent? One person asked here, "With what?" That's the point. We can see that it doesn't contradict itself, but we really can't tell if it doesn't contradict any other stated requirement. So just this example by itself, we can't tell if it's consistent, but we'll just assume it is for the point of this example. Is it unambiguous, susceptible to only one interpretation? Yes. And it is verifiable? Very good. It has been expressed in a way that can be tested.

So here we have another question. "It is said that requirements define the detailed"—and I'm going to give you a hint again—who, what, or how? Okay. Somebody else want to make a guess? Well, it is "what." It's still a what. We have the high-level what in the user needs, but in this case we're talking about the detailed what, because we still haven't been talking about the how. We're not doing the design. So that was a little trick question and hopefully everybody learned something from that.

This is our next learning objective, and before I get into this, are there any other questions about the previous learning objective? Okay. Now we're going to go into defining the system and interfaces as a functional architecture. First we're going to talk about context diagrams and then we're going to talk about the functional architecture. The main point of this section is that good pictures are essential.

The context diagrams, they show what's inside and outside of the system, basically. It shows the boundary that defines where the external interfaces are. And believe it or not, it's often the most difficult and critical task for a project, and I've been on projects and can vouch for this, that defining exactly where those boundaries are can be difficult. That's because it requires skill and creativity to explore alternative possibilities, and especially if you have a group project, you're going to get a lot of opinions about that. But the other issue about this is that all the later work on the project is affected by where you draw that line. So I suggest that you use a lot of graph paper. Again, another way of saying what I just said, that you don't know what the system has to do until you have the boundary, and you don't have the boundary until you know what the system has to do. So again, use of graph paper is very helpful when it comes to developing your context diagrams.

Here's an example, a system context diagram for an Amber Alert system. Now, some of you on this phone call may be experts in this area so please bear with me. This is just a simple diagram I put together for the purposes of the course. Here you see the Amber Alert system is what we're building, and these other boxes around it, in this case, are outside the system, but we interface to those. Now, sometimes what you can do is also, in a context diagram, it will have an in-and-out table, and I've used those before also, especially when you have a multi-headed development team, where people kind of forget what decisions were made, it's helpful to have an in-and-out

table that you record down what's in the system and what's out of the system, along with the diagram.

If the boundary is uncertain, then you need to put more effort into defining it. If you defined your boundary but most of the system work is repeatedly crossing the boundary to achieve anything, then you might want to think about where that boundary should be. And the truth of the matter is, in many cases it's a business choice, not a technical one, whether something is inside the system boundary or not. A good test, also, is to think through, when you're doing your diagram here, is it true for all the important stakeholders? That's a wishy-washy word, "important," but we know people have influence in different systems, but is it true for the important stakeholders? That's a good sign to move on.

Functional architectures describe the functions within a system and what comes into and out of the functions. We don't want to confuse a functional architecture with defining a particular device or subsystem, although often they will lead to that. We use this term "function elements" so that we try and stay clear of that. It's a detailed design drawing. The labels on the drawing and on the lines of communication, they generally tell the story, and the structure of this diagram then helps us to articulate the user needs and organize the requirements. I can tell you, on just about every project I've worked on, that drawing the picture first and getting the picture right makes organizing and developing the needs and requirements much easier.

We already looked at the context diagram. Often a context diagram will show up in the concept of operations. And then we get into the functional architecture, and that will show up at a requirements level. Functional architecture can be drawn in many ways. This is just one of the ways, and I'll step through this. Actually, in this case, we have these boundaries that were in our circle before, in the context diagram. Here we have these boundaries drawn by dotted lines, and our system is broken up into these four functional elements. We have emergency processing, police dispatching, traffic management ops, and highway patrol dispatching.

I forgot to put on the labels, so here we are. So we see a caller will make a verbal report to the emergency message processing which sends an incident report to the police dispatching, who sends it to the different other elements – the traffic management operations and the highway patrol – and the highway patrol and police talk to their vehicles. The traffic management operations send it to the radio stations and put the Amber Alert up on the freeway signs.

Any questions about this? Again, this is just one way of doing a functional architecture and, in fact, I believe in the Supplement I've given some other examples. Now let's talk about decomposition. This is our learning objective number four, using decomposition of the architecture and requirements to define the system. We'll talk about decomposition of the architecture and then we'll talk about decomposition of the requirements.

So here is the picture that we had just spoken about in the previous learning objective. I've taken off the labels just for simplicity, because we have a little animation in here. You notice the traffic management section is highlighted. That's maybe, for some of us on the phone, that's all we would really be worrying about is that circle. But in this system as developed—our user needs and requirements—we found out that traffic management operations actually broke up into these

three sub-elements: media dispatching, traffic control operations, and DMS control. So we had these other elements – the DMS control system, I know in one city was a separate system from their main traffic control operations, so this would be that kind of illustration.

In turn, just as there is a decomposition of the architecture, then there is decomposition of the requirements. Now, note that this requirement referred to the first picture we had, that had traffic management operations. "Traffic management operations shall notify the public of an Amber Alert." This might work if traffic management operations was a single subsystem, but since we've shown that we broke that down, in this case we have two requirements that are used, because of that decomposition. So "Traffic control operations shall send an Amber Alert notification to media dispatching," and "Media dispatching shall send an Amber Alert notification to the radio stations." In both cases, these requirements could trace to the previous requirement or you may find that you bring these up a level sometimes—it depends on the project—where all of these would eventually trace to the user need that was expressed up in the concept of operations.

Again, so these requirements refer to the sub-element, traffic control operations. The previous requirement referred to traffic management operations. And again, when we're developing requirements, you'd still want to look at the functional performance, non-functional, and constraints at each subsystem level. So we would look at those things in an iterative process as we go through our decomposition. We'd go back to what we learned previously about the different kinds of requirements that we might need at every level.

Now we'll move on to learning objective five. Well, let me ask again. Are there any questions about our previous learning objective, decomposition? Okay. Now we're going to look at verifying that requirements are complete and correct, and we'll talk first about correctness, we'll talk about completeness, we'll talk about using traceability. Usually we talk about verifying correctness and validating completeness. Traceability is a tool that we use to help us do both of those things.

This is a friendly slide telling you that we're going to have an activity, but we've been having activity throughout this so it should be no surprise. Let's take a look at verifying that the requirements are correct. When we do that, we apply the rules that we had learned previously. We have our structure of our well-formed requirement, and then we want to also look at the rest of our characteristics. So in this case what's the actor? Traffic control operations. Okay. What's the action? Acknowledge. Right. And what's the target? The target is the receipt.

Okay, what's interesting here is I don't see a constraint or a localization, and it sort of feels like it doesn't have quite the impact, doesn't it? So it is necessary? Most likely. It looks very concise. Is it attainable? Certainly it seems that way. Standalone? Well, it looks like it, although it would have to be in some section that all the terms that are in here are clear. Again, is it consistent? We know it's consistent with itself, but again, we don't know if it's consistent with the rest of the requirements.

Now, why might this be ambiguous? Yeah, we don't know how this acknowledgement is to be conducted. We've got some argument, some people saying, "Well, it's not ambiguous because

there isn't enough information there to tie it down." In my opinion, I think it's ambiguous because it needs some sort of localization. From where? Is it a message or is it verbal? Again, you can get experienced people that will argue about these things and sometimes they come to matters of opinion, but I certainly think that some sort of localization in here would help with any ambiguity. But one way or another, it's probably verifiable, right? It's verifiable that one way or another we'd find some way of doing it.

Okay. Validating requirements are complete. Now, when we talk about decomposition, we sometimes use a parent-child terminology, and we'll use that here, too. So a user need may be a parent of a systems requirement. A systems requirement may be a parent of a subsystem requirement. A systems requirement will be a child of a user need. And a subsystem requirement will be a child of a systems requirement. So when we're talking about whether requirements are complete, we need to make sure that all aspects of the user needs or parent requirement are addressed by the child requirements.

We also need to make sure that the requirements at a given level are consistent with each other. I think that's one of the hardest parts, as we step through things, is that we make sure that we keep in the same context when we're expressing requirements so that they stay at the same level. Then again, as I said, traceability is an important tool for checking completeness and correctness, and we'll talk more about that in a moment.

Now we're going to talk about need-to-requirement local consistency. Here is our need: Extreme Temperatures and Humidity – the user needs the TFCS to operate under extreme hot, cold, and human environmental conditions. That's a good user need. We don't have the rationale here; pretend we understand that. So in logical consistency, what we try to do is we're trying to make sure that all aspects of the user parent requirement are addressed in the child requirement.

Here is some interactivity opportunity for you all. Where might you see some inconsistencies or logical inconsistencies from between this need and this requirement? The requirement says the TFCS shall be capable of withstanding an ambient storage temperature range of -45 degrees Celsius to +85 degrees Celsius. There are a couple of them actually.

If this is the one need and this is the one requirement expressed for this need, what is logically inconsistent about this requirement to this need? Well, I'll give you this first one. This requirement doesn't speak to humidity. It said "under extreme hot, cold, and human environmental conditions." Well, this requirement speaks only to temperature, so that would be considered a logical inconsistency. There is one other one I wanted to mention. Can you see it? Okay. I'll give this one to you, too. This one, "the user needs the TFCS to operate under extreme hot, cold, and humid environmental conditions."

Hey, you know what? I hadn't scrolled down in my chat and seen some of the answers, and the class actually answered these questions. The need speaks to operational aspects and a requirement speaks to storage, so thank you and my apologies. Let's take a look at another one.

Now, that was a logical inconsistency between the user need and a requirement. Here are two requirements. What is inconsistent in these two requirements? And I'm going to just let you read

them yourself. Well, actually, for the sake of the presentations, I'll read it. “The output assembly shall accept switch pack modules suitable for controlling field displays that operate at nominal 120 VAC 60Hz.” The second requirement was “The output assembly shall accept switch packs suitable for controlling field displays that operate at 48 VDC (+/- 2.0 VDC).”

These are at the same level. There are two devices or two things being talked about here, and what might be inconsistent about it? Well, actually, these look like they're two different parts. The question was, “Does it have to do with AC and DC?” Well, these could be two different components of the system. But here, someone answered, very good. They mentioned modules versus switch packs. Here's the terminology for the same kind of device but in the first one they called it switch pack modules and in the second requirement they called it switch packs. That's great. That's an inconsistency.

Then, another thing—this was very insightful—was the use of nominal voltages in the first one, but then in the second one they gave a voltage and a range, so that was inconsistent. Someone asked “What is low voltage?” Well, the requirement in this case is called low voltage and they define what low voltage is in the requirement. I happen to know where this came from and they're looking at voltages that are low enough so that if you have a signal head knocked down during a storm or something, the voltages will be still low enough so that they're safe. That goes to the user need.

Now we'll talk about using traceability. Traceability is a tool to help us with completeness and correctness. It mainly verifies completeness but also helps with correctness, in other words, understanding that it's necessary. The second bullet was, we need to know that every need is a parent, so there is at least one requirement for it. There are no orphan requirements—that's the next one—so there's no requirement that can't be traced to at least one need or one superior requirement. And the consequence of the first thing is that any need that is not addressed by at least one requirement means that a requirement was missed, or we need to reevaluate that user need. Does that make sense?

If there isn't a requirement, then we need to say, “Oh, we forgot this,” or “Do we really need this need?” In turn, every requirement that does not address at least one need means that the requirement must be reevaluated or a user need was missed, and that is often the case. As we go to developing, throughout our systems development process, we find out that, well, we forgot to put this in because this is clearly a requirement for our system but we didn't actually express the need in our previous stage of development. What you find is there is an iterative aspect to this, and I didn't say this when we were going through the systems life cycle but the systems life cycle is definitely an iterative process. As you go down through the stages, you find things that you missed and you correct and you move forward. So every aspect of each user need should be addressed in the requirements.

There are lots of tools out there to help us. It's often the case that we can handle this in some sort of document form, but there are tools out there that can help with the Systems Engineering Process, especially if you get into a very large system, that becomes more necessary. And some of the tools will express this traceability graphically. You'll have some sort of graphical representation of the need to the requirement. This is just an example of that. We have “manage

the real-time clock" as a user need, apparently, in this document, in Chapter Two, and then down in Chapter Three, as well, the requirements are, and you see these four requirements being used to define this user need.

In the documents, especially in our communication standard, we will use various kinds of matrices. This one is called a Needs-to-Requirements Traceability Matrix, and what you see is a user need identifier, the name of the user need in this table, and then we have the requirement identifiers listed and the requirement names. This is a way of expressing in a tabular form. And again, a little refresher here. What kind of relationship is this? Is this a one-to-one relationship or what? What kind of relationship is this, of a user need-to-requirement? Let's see if you remember the term. Well, one-to-four, that was good. What we called it was one-to-many. Yeah, one-to-many. There we go. Good.

So I won't go down through all of these. I'll leave that for you to read. Here is the user need that we used, and then here are the requirements that reflected the requirements that were expressed in the table. There's "Set Date and Time" and "Set Daylight Saving Time."

Now I have this slide called Traceability Beyond Requirements, and that's because it's a tool. You saw it as a tool between user needs and requirements, and requirements to other requirements, but it's also a tool that can be extended to the design. Once you get into designing the product or system, you can find that, hey, you can trace those design elements back to the requirements and then back to the needs. So it's very powerful and helps someone trying to understand the system to be able to see that.

And then, in turn, it also helps you with testing. Remember we talked about requirements-driven testing? This traceability from a requirement through the design documents and such will help us in our testing, and actually through the development of the product. It will help us with system acceptance and validation. And then this traceability can also help us with procurements, because you may have features that you trace that you want, and maybe there are other features that you trace that you don't want.

So we would say that every aspect of each requirement should be addressed in the design. We'd say every aspect of each requirement should be addressed by a test procedure, and then use the traceability to help us with our system acceptance and validation. We use this traceability in procurements with our communications standards. We use traceability to help us in procurements to allow end users to select the features that they want in their system, and we'll get more into that in the next module, A203.

Okay. Any questions about this? Okay. We'll go on to our last learning objective, and that is applying what we learned to ITS communications standards. Up until now we've talked about requirements development in general, and now we're going to get a little more specific to the focus of this program, and then we'll also look at a preview of other modules to come.

The use of standards—we showed you requirements and high-level design and things in the systems life cycle—but the use of communications standards usually starts in the design phases of the system. So if you think of a large system, you might not be that concerned about the

specific communications down at lower levels of this system that you have yet to define. So it's usually in the design phases of the system that we are concerned with, with the communications standards. And they're typically considered part of subsystems development. Now, what we've done is we've applied the Systems Engineering Process to standards development to the standard's content.

We have found, in developing the standards, by doing this, that we get complete and correct—at least to the best of our ability—standards. If we don't think about that as we proceed through our standards development, we find that the standards come back with a lot of issues that we didn't cover because they weren't discovered during the development of that standard.

And again, here is our systems life cycle, and this just kind of illustrates it. This is a subsystem. It's part of that subsystem development that this is where we look at standards, and that's all this graphic is, and this is an example of the transportation sensor systems standards of NTCIP. I happened to work on that one so it got into my example. So we sometimes call this "the little V," and so depending on the size of your project, you may have just the big V, but often systems projects end up with a little V for the subsystems themselves.

Now we'll talk about the content of center-to-field communication standards that do include Systems Engineering Process information. They usually have a general section, introductory section, they have a concept of operations where the user needs are. They have functional requirements. Then they have design details. In communications standards we're talking about dialogs—that's the exchange of information or the ordering of the exchange of information—and then what we call the object definitions—that's defining the data elements that are being communicated. And then there are annexes where we have things like traceability matrices, test procedures, and documentation of the revisions.

So what you see here is the Systems Engineering Process applied to standards development, and it makes it clear to the user, to the person using this document, what the thought process was and what the development process was by the developers of this standard.

Often there is a Requirements Traceability Matrix. This is what we call this. And here again you have the requirements ID, you have the requirement, we have the dialog ID—and again, the dialog occurs when you need to have a certain ordering of the messages being between two devices or between the central system and the device, to achieve the operations. That's called a dialog. And then we have the object ID, which refers to the different data elements within the standard. Again, for instance, if, in a standard, a certain set of requirements or set of needs are considered optional, then this kind of traceability would allow a user to find out, hey, my system doesn't need to support these items.

And these are the modules that we have on standards that include SEP content. We have that for dynamic message signs, we have it for environmental sensors, we have it for the traffic management data dictionary. That is a center-to-center standard.

Now, there are plenty of standards out there that do not have SEP content, and this is a kind of outline that is often found in these standards. There's an overview and some very brief general

information, and then it goes right into the definition of the data items within the standard, and that's done in a management information base, or called a MIB. And then there are usually conformance groups. That's kind of a grouping of the data elements that work together. For instance, in the actuated signal control, there is a phase conformance group and others that all the object definitions have to do with phases are grouped together. All those objects that are having to do with coordination are grouped together. Things like that. And then there's a conformance statement.

What we're suggesting in this program is that even though these definitions are there, on a given project we're recommending that you write the needs and requirements for your project and then do it in a language that is based on the standard. And, I want to say something else. Just because these standards don't have the Systems Engineering Process doesn't make them incorrect or bad. It's just that they don't have that kind of information that helps you as much as having it in there.

In A203, we'll review some of the things that we did here in this course, and that's the next module. We'll go into some other items that you need to know before writing requirements, and then we'll actually do a little bit of requirements writing for these standards that don't have SEP content. We'll just see that examples that we go through there, so that you'll be able to do that on your own. Any questions about this last learning objective, applying the Systems Engineering Process to our communication standards?

Okay. What did we learn today? We learned to define requirements for overall operation to satisfy user needs. We learned the concept of a well-formed requirement. We learned to define the system and interfaces as a functional architecture. We learned to use decomposition of the architecture and requirements as necessary to properly define the system. We learned to verify that requirements are complete and correct. And we also learned how requirements development applies to ITS communication standards.

Hopefully we got that out of our course today, and I'd like to just open up to see if there are any other questions at this point. Okay. If there are no more questions, this now ends this training module, A103. Thank you.

*[End of Audio]*